REMARKS

Objection to the Claims

The Office Action objects to the claims for several informalities, and correction has been provided. The "Deleted" text in Claims 1, 8 and 9 has been deleted. Claims 5, 13 and 14 now has the status shown "Original." Claim 8 has been amended as intended in the last office action response.

Rejection under 35 U.S.C. § 112, first paragraph

Claim 17 has been amended to correct one the two step d) to step e). Claim 17 was appropriately amended as well.

Rejections under 35 U.S.C. §102

The Office Action rejects Claims 1-5, 11-13 and 15-19 as anticipated by Gere US2004/0019778. Applicant respectfully traverses this rejection.

As is well known, an anticipation rejection requires that "each and every element as set forth in the claim is found, either expressly or inherently described, in a single piece prior art reference." Verdegaal Bros. v. Union Oil Co. of California, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Gere does not anticipate Applicant's amended claim 1 which recites the limitations of "reconfiguring the second computer by merging configuration and program information of said first computer into a local system environment of said second computer; and thereafter operating said second computer in accordance with said settings and under an operating system which was already installed on the second computer without performing a system reboot."

Gere's method transfers a bootable operating system from a first machine to a second machine. In Gere, the entire operating system is captured on a portable drive, and after the transfer of the operating system is complete from a first machine to a second machine, reconfiguration of the second machine occurs upon the booting of the transferred operating system. **Gere always requires a system reboot of the computer and transfer the operating system**. Amended Claim 1 specifically teaches against this.

By contrast, Applicant discloses a unique method of transferring a portable operating environment – not the operating system. Applicant's method involves a transfer of configuration, program/program data, and dynamic link library information from a first computer operating environment to a second computer operating environment via a portable data medium and requires a Portable Operating Environment software utility. The reconfiguration of a second environment occurs upon transfer of the information and **does not require a reboot**. **Applicant's method does not transfer an entire operating system, which can never be done without a system reboot.** This is an important distinction between Gere and the invention which the Examiner has failed to address in his analysis. The amended claim specifically claims this feature, and unless Gere teaches this limitation a § 102 rejection cannot be sustained. Because Applicant's method does not require booting of the second system environment prior to resuming operating, it is not suggested, taught, or disclosed by Gere.

Furthermore, Claim 1 requires a utility program. Gere does not suggest, teach, or disclose such a utility program. Because Gere fails to teach this limitation, a § 102 rejection cannot be sustained.

For at least the foregoing reasons, Applicant asserts that Gere does not anticipate Claim 1. Claims 2-16 depend from Claim 1 and are likewise not anticipated by this reference and further add additional limitations to the allowable Claim 1. Therefore, Applicant respectfully requests that this rejection be withdrawn and claims 1 - 16 be allowed.

The Examiner also rejected Claims 17-19 under §102 citing Gere. Claim 17 pertains to registry information being transferred from a first computer to a second computer. Registry information **is not** part of the operating system. The registry data comprises settings and options for the operating system to use and is not part of the operating system. *See Exhibit 1, Wikipedia printout; and Spec., para 0033 and Fig. 5.* Therefore, copying and moving the operating system cannot read upon Claim 17, which involves only the registry data.

Furthermore, Claim 17 requires using a first and second copy of a configuration utility to capture and merge the registry information. Gere does not suggest, teach, or disclose such configuration utility. Because Gere fails to teach this limitation, a § 102 rejection cannot be sustained.

For at least the foregoing reasons, Applicant asserts that Gere does not anticipate Claim 17. Claims 18-20 depend from claim 17 and are likewise not anticipated by this reference and further add additional limitations to the allowable Claim 17. Therefore, Applicant respectfully requests that this rejection be withdrawn and claims 17 - 20 be allowed.

In order to establish a prima facie case of obviousness, the prior art reference or references, *inter alia,* must teach or suggest all the claim limitations. See *Manual of Patent Examining Procedure*, § 2143, Revision 5, August 2006. Applicant asserts that neither Gere nor Hamilton, alone or in combination teaches or suggests Applicant's claims as found in Claims 1 and 17. Since, neither Gere nor Hamilton teaches or suggests the limitation of Applicant's Claim 1 or Claim 17, the rejections those claims and their dependent claims are traversed and the claims 1-20 allowable.

CONCLUSION

Based on the foregoing amendment and remarks, reconsideration of the application is respectfully requested. All objection and rejections have been traversed or accommodated. This response is intended to be a complete response to the Office Action mailed August 24, 2007. No new matter has been added. Applicants submit that the Claims are now in a condition for allowance, and such notice is hereby earnestly solicited.

February 24, 2008

Respectfully submitted,

Malcolm Pipes

Malcolm Pipes, Reg. No. 46,995
Attorney for Applicant

GROOVER & HOLMES
Customer Number 29106
P.O. Box 802889
Dallas, Texas 75380
Telephone: (972) 980-5840
Facsimile: (972) 980-5841

Windows Registry

From Wikipedia, the free encyclopedia

The **Windows registry** is a directory which stores settings and options for the operating system for Microsoft Windows 32-bit versions, 64-bit versions and Windows Mobile. It contains information and settings for all the hardware, operating system software, most non-operating system software, users, preferences of the PC, etc. Whenever a user makes changes to Control Panel settings, file associations, system policies, or most installed software, the changes are reflected and stored in the registry. The registry also provides a window into the operation of the kernel, exposing runtime information such as performance counters and currently active hardware. This use of registry mechanism is conceptually similar to the way that Sysfs and procfs expose runtime information through the file system (traditionally viewed as a place for permanent storage), though the information made available by each of them differs tremendously.

The *Windows registry* was introduced to tidy up the profusion of per-program INI files that had previously been used to store configuration settings for Windows programs.[1] These files tended to be scattered all over the system, which made them difficult to track.

Contents

Structure

Keys and Values

The registry contains two basic elements: keys and values.

Registry **Keys** are similar to folders - in addition to values, each key can contain subkeys, which may contain further subkeys, and so on. Keys are referenced with a syntax similar to Windows' path names, using backslashes to indicate levels of hierarchy. E.g. HKEY_LOCAL_MACHINE\Software\Microsoft\Windows refers to the subkey "Windows" of the subkey "Microsoft" of the subkey "Software" of the HKEY_LOCAL_MACHINE key.

Registry **Values** are name/data pairs stored within keys. Values are referenced separately from keys. Value names can contain backslashes but doing so makes them difficult to distinguish from their key paths. The Windows API functions that query and manipulate registry values take value names separately from the key path and/or handle that identifies the parent key.

The terminology is somewhat misleading, as the values are similar to an associative array, where standard terminology would refer to the name part of the value as a "key". The terms are a holdout from the 16-bit registry in Windows 3, in which keys could not contain arbitrary name/data pairs, but rather contained only one unnamed value (which had to be a string). In this sense, the entire registry was like an associative array where the keys (in both the registry sense and dictionary sense) formed a hierarchy, and the values were all strings. When the 32-bit registry was created, so was the additional capability of creating multiple named values per key, and the meanings of the names were somewhat distorted[2].

There are a number of different types of values:

List of Registry Value Types

0	REG_NONE	No type
1	REG_SZ	A constant string value
2	REG_EXPAND_SZ	An "expandable" string value that can contain environment variables
3	REG_BINARY	Binary data (any arbitrary data)
4	REG_DWORD/REG_DWORD_LITTLE_ENDIAN	A DWORD value, a 32-bit unsigned integer (numbers between 0 and 4,294,967,295 [2^{32} – 1]) (little-endian)
5	REG_DWORD_BIG_ENDIAN	A DWORD value, a 32-bit unsigned integer (numbers between 0 and 4,294,967,295 [2^{32} – 1]) (big-endian)
6	REG_LINK	symbolic link (UNICODE)
7	REG_MULTI_SZ	A multi-string value, which is an array of strings
8	REG_RESOURCE_LIST	Resource list
9	REG_FULL_RESOURCE_DESCRIPTOR	Resource descriptor
10	REG_RESOURCE_REQUIREMENTS_LIST	Resource Requirements List
11	REG_QWORD/REG_QWORD_LITTLE_ENDIAN	A QWORD value, a 64-bit integer (either big- or little-endian, or unspecified)

Hives

The Registry is split into a number of logical sections, or "hives".[3] Hives are generally named by their Windows API definitions, which all begin "HKEY". They are abbreviated to a three- or four-letter short name starting with "HK" (e.g. HKCU and HKLM).

The HKEY_LOCAL_MACHINE and HKEY_CURRENT_USER nodes have a similar structure to each other; applications typically look up their settings by first checking for them in "HKEY_CURRENT_USER\Software\Vendor's name\Application's name\Version\Setting name", and if the setting is not found looking instead in the same location under the HKEY_LOCAL_MACHINE key. When writing settings back, the reverse approach is used — HKEY_LOCAL_MACHINE is written first, but if that cannot be written to (which is usually the case if the logged-in user is not an administrator), the setting is stored in HKEY_CURRENT_USER instead.

HKEY_CLASSES_ROOT

Abbreviated HKCR, HKEY_CLASSES_ROOT stores information about registered applications, such as Associations from File Extensions and OLE Object Class IDs tying them to the applications used to handle these items. On Windows 2000 and above, HKCR is a compilation of HKCU\Software\Classes and HKLM\Software\Classes. If a given value exists in both of the subkeys above, the one in HKCU\Software\Classes is used.[4]

HKEY_CURRENT_USER (HKCU)

Abbreviated HKCU, HKEY_CURRENT_USER stores settings that are specific to the currently logged-in user. The HKCU key is a link to the subkey of HKEY_USERS that corresponds to the user; the same information is reflected in both locations. On Windows-NT based systems, each user's settings are stored in their own files called NTUSER.DAT and USRCLASS.DAT inside their own Documents and Settings subfolder.

HKEY_LOCAL_MACHINE (HKLM)

Abbreviated HKLM, HKEY_LOCAL_MACHINE stores settings that are general to all users on the computer. On NT-based versions of Windows, HKLM contains four subkeys, SAM, SECURITY, SOFTWARE and SYSTEM, that are found within their respective files located in the %SystemRoot%\System32\Config folder. A fifth subkey, HARDWARE, is volatile and is created dynamically, and as such is not stored in a file. Information about system hardware drivers and services are located under the SYSTEM subkey, while the SOFTWARE subkey contains software and Windows settings.

HKEY_USERS

Abbreviated HKU, HKEY_USERS contains subkeys corresponding to the HKEY_CURRENT_USER keys for each user registered on the machine.

HKEY_CURRENT_CONFIG

Abbreviated HKCC, HKEY_CURRENT_CONFIG contains information gathered at runtime; information stored in this key is not permanently stored on disk, but rather regenerated at boot time.

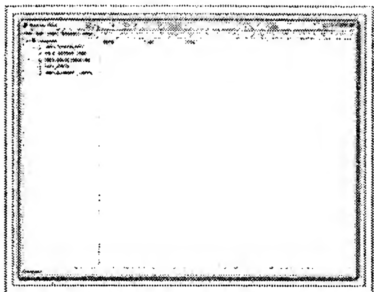
HKEY_PERFORMANCE_DATA

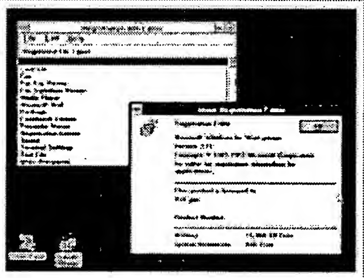
This key provides runtime information into performance data provided by either the NT kernel itself or other programs that provide performance data. This key is not displayed in the Registry Editor, but it is visible through the registry functions in the Windows API.

Editing

Manual editing

The registry can be edited manually in Microsoft Windows by running regedit.exe or regedt32.exe in the Windows directory. However, careless registry editing can cause irreversible damage. Thus, performing back-ups of the registry is highly recommended. Many optimization and "hacking" tools are available to modify this portion of the Windows operating system; it is preferable not to use them unless one has a knowledge of registry workings or wishes to learn more about the registry.


The *Registry Editor* in Windows Vista


Windows 3.11 Registry Editor

A simple implementation of the current registry tool appeared in Windows 3.x, called the "Registration Info Editor" or "Registration Editor". This was basically just a database of applications used to edit embedded OLE objects in documents.

Windows NT introduced permissions for Registry editing. Windows NT 4 and Windows 2000 were distributed with both the Windows 9x REGEDIT.EXE program and Windows NT 3.x's REGEDT32.EXE program. There are several differences between the two editors on these platforms:

- REGEDIT.EXE had a left-side tree view that began at "My Computer" and listed all loaded hives. REGEDT32.EXE had a left-side tree view, but each hive had its own window, so the tree displayed only keys.
- REGEDIT.EXE represented the three components of a value (its name, type, and data) as separate columns of a table. REGEDT32.EXE represented them as a list of strings.
- REGEDIT.EXE supported right-clicking of entries in a tree view to adjust properties and other settings. REGEDT32.EXE required all actions to be performed from the top menu bar.
- REGEDIT.EXE supported searching for key names, values, or data throughout the entire registry, whereas REGEDT32.EXE only supported searching for key names in one hive at a time.
- Because REGEDIT.EXE was directly ported from Windows 95, it did not support permission editing (permissions do not exist on Windows 9x). Therefore, the only way to access the full functionality of an NT registry was with REGEDT32.EXE.
- REGEDIT.EXE only supported string (REG_SZ), binary (REG_BINARY), and DWORD (REG_DWORD) values. REGEDT32.EXE supports those, plus expandable string (REG_EXPAND_SZ) and multi-string (REG_MULTI_SZ). Attempting to edit unsupported key types with REGEDIT.EXE on Windows 2000 or Windows NT 4 will result in conversion to a supported type that cannot be reversed.[5]

Windows XP was the first system to integrate these two programs into one, adopting the old

REGEDIT.EXE interface and adding the REGEDT32.EXE functionality. The differences listed above are not applicable on Windows XP and newer systems; REGEDIT.EXE is the improved editor, and REGEDT32.EXE is simply a stub that invokes REGEDIT.EXE.

Command line editing

The registry can be manipulated from the command line with the reg.exe utility. It is included in Windows XP and Windows Vista and can be downloaded separately for previous versions. Alternative locations include the Resource Kit CD's or the original Installation CD of Windows.

```
reg.exe Operation [Parameter List]
```

```
Operation [QUERY|ADD|DELETE|COPY|SAVE|LOAD|UNLOAD|RESTORE|COMPARE|EXPORT|IMPORT]
```

Also, a .reg file (a text-based human-readable file format for storing portions of the registry) can be imported from the command line with the following command:

```
regedit.exe /s file
```

The /s means the file will be *silent merged* to the Registry. If the /s parameter is omitted the user will be asked to confirm the operation. In Windows 98 and Windows 95 the /s switch also caused regedit.exe to ignore the setting in the registry that allows administrators to disable it. When using the /s switch Regedit does not return an appropriate return code if the operation fails, unlike reg.exe which does. This makes it hard to script, however a possible workaround is to add the following lines into your batch file:

```
regedit /s file.reg
regedit /e test.reg "key"
if not exist test.reg goto REGERROR
del test.reg
```

The default association for .reg files in many versions of Microsoft Windows, starting with Windows 98 does require the user to confirm the merging to avoid user mistake.

Registry permissions can be manipulated through the command line using the `SubInACL.exe` tool. The permissions on the HKEY_LOCAL_MACHINE\SOFTWARE key can be displayed using:

```
subinacl /keyreg HKEY_LOCAL_MACHINE\software /display
```

To set the owner of the key HKEY_LOCAL_MACHINE\software and all of its subkeys to Administrator:

```
subinacl /keyreg HKEY_LOCAL_MACHINE\software /setowner=Administrator
subinacl /subkeyreg HKEY_LOCAL_MACHINE\software /setowner=Administrator
```

To grant full access rights to the HKEY_LOCAL_MACHINE\software key to Administrator:

```
subinacl /keyreg HKEY_LOCAL_MACHINE\software /grant=Administrator=F
```

Programs or scripts

The registry can be edited through the APIs of the Advanced Windows 32 Base API Library (advapi32.dll).[6]

List of Registry API functions			
RegCloseKey	RegOpenKey	RegConnectRegistry	RegOpenKeyEx
RegCreateKey	RegQueryInfoKey	RegCreateKeyEx	RegQueryMultipleValues
RegDeleteKey	RegQueryValue	RegDeleteValue	RegQueryValueEx
RegEnumKey	RegReplaceKey	RegEnumKeyEx	RegRestoreKey
RegEnumValue	RegSaveKey	RegFlushKey	RegSetKeySecurity
RegGetKeySecurity	RegSetValue	RegLoadKey	RegSetValueEx
	RegNotifyChangeKeyValue	RegUnLoadKey	

Some programming languages, like Visual Basic, offer built-in runtime library functions that enable programs to store settings in the registry. Another way is to use the Windows Support Tool Reg.exe by executing it from code.[7]

Many scripting languages such as Perl (with `Win32::TieRegistry`) and VBScript also enable registry editing from scripts.

Locations

The Registry is stored in several files; depending upon the version of Windows, there will be different files and different locations for these files, but they are all on the local machine, except for the `Ntuser.dat` files. There is one such file per user that contains the information in HKEY_CURRENT_USER; it may be placed on another computer to allow for roaming profiles. The policy file, which is usually stored on a server in the local network, may also be located remotely.

Windows NT, 2000, XP, Server 2003, and Vista

The following Registry files are stored in `%SystemRoot%\System32\Config\`:

- `Sam` – HKEY_LOCAL_MACHINE\SAM
- `Security` – HKEY_LOCAL_MACHINE\SECURITY
- `Software` – HKEY_LOCAL_MACHINE\SOFTWARE
- `System` – HKEY_LOCAL_MACHINE\SYSTEM
- `Default` – HKEY_USERS\.DEFAULT
- `Userdiff` - Not associated with a hive. Used only when upgrading operating systems.[8]

The following files are stored in each user's profile folder:

- `%UserProfile%\Ntuser.dat` – HKEY_USERS\<User SID> (linked to by HKEY_CURRENT_USER)

- `%UserProfile%\Local Settings\Application Data\Microsoft\Windows\Usrclass.dat` (path is localized) – HKEY_USERS\<User SID>_Classes (HKEY_CURRENT_USER\Software\Classes)

Windows 95, 98, and Me

The registry files are named `USER.DAT` and `SYSTEM.DAT` and are stored in the `%WINDIR%` directory. In Windows Me, `Classes.dat` was added. Also, each user profile (if profiles are enabled) has its own `USER.DAT` in profile's directory.

Windows 3.11

The registry file is called `Reg.dat` and is stored in the `C:\WINDOWS` directory.

Policy files

Since Windows 95, administrators can use a special file to be merged into the registry, a policy file. The policy file allows administrators to prevent non-administrator users from changing registry settings like, for instance, the security level of IE and the desktop background wallpaper. The policy file is primarily used in a business with a large number of computers where the business needs to be protected from the users and the users need to be protected from themselves.

The default extension for the policy file is `.pol`. The policy file filters the settings it enforces by user and by group (a "group" is a defined set of users). To do that the policy file merges into the registry, preventing users from circumventing it by simply changing back the settings. The policy file is usually distributed through a LAN, but can be placed on the local computer.

Policy file editor

The policy file is created by a free tool by Microsoft that goes by the filename `poledit.exe` for Windows 95/Windows 98 and with a computer management module for NT-based systems. The module will not work in Windows XP Home Edition, but it does work in the Professional edition with filename Gpedit.msc. The editor requires administrative permissions to be run on systems that uses permissions. The editor can also directly change the current registry settings of the local computer and if the remote registry service is installed and started on another computer it can also change the registry on that computer. The policy editor loads the settings it can change from `.adm` files, of which one is included, that contains the settings the Windows shell provides. The `.adm` file is plain text and supports easy localisation by allowing all the strings to be stored in one place. The policy editor has been renamed to Group Policies in newer versions of Windows.

Advantages

Changing from having one or more INI files per program to one centralized registry has its good points:

- The registry keeps machine configuration separate from user configuration. When a user logs into a Windows NT/2000/XP/Server 2003 computer, the user-based registry settings are loaded from a different path than the system wide settings. This allows programs to more easily keep per-user configuration, as they can just work with the "current user" key, whereas in the past they tended to just keep system-wide per-program settings.
- Group Policy allows administrators on a Windows-based computer network to centrally manage

program and policy settings. Part of this involves being able to set what an entry in the registry will be for all the computers on the network, and affect nearly any installed program — something almost impossible with per-program configuration files each with custom layouts, stored in dispersed locations.
- Because the registry is accessed through a special API, it is available to scripts and remote management using WMI. Each script does not have to be customized for every application's unique configuration file layouts and restrictions.
- The registry can be accessed as one item over a network connection for remote management/support, including from scripts, using the standard API.
- It can be backed up more easily, in that it is just a small number of files in specific locations.
- Portions of settings like any subset of an application configuration can be saved in a text-based .REG file, which can be edited with any text editor later. .REG files can easily be merged back into the registry both by unattended batch file or by the user just double-clicking on the file without harming any setting that is not explicitly stated in the .REG file. This is very useful for administrators and support personnel who want to pre-set or pre-configure only a few options like approving the EULA of a certain application.
- Since accessing the registry does not require parsing, it can be read from and written to more quickly than a text file can be.
- Registry changes and readings can be tracked via a tool like Winternals' RegMon on value level. This is a big advantage for generating scripts in networks as well as debugging problems.
- Registry keys are independent of the Windows language, the Windows installation drive and path and even the Windows versions as such. So support personnel can easily give out one set of instructions, without having to handle these things, unlike, for example, files in the user profile, which can be on different paths on each installation.
- The registry is constructed as a database, and offers DB-like features such as atomic updates. If two processes attempt to update the same registry value at the same time, one process's change will precede the other's, so one will only last a short time until the second gets written. With changes in a file system, such race conditions can result in interleaved data that doesn't match either attempted update. Windows Vista provides transactional updates to the registry, so the atomicity guarantees can be extended across multiple key and/or value changes, which traditional commit-abort semantics. (Note that NTFS provides such support for the file system as well, so the same guarantees could be obtained with traditional configuration files.)

Disadvantages

However, the centralized Registry introduces some problems as well:

- The HKEY_LOCAL_MACHINE part is a single point of failure — damage to the Registry can render a Windows system unbootable, in extreme cases to a point that cannot be fixed, and requires a full reinstall of Windows. Windows 2000 (including server) keeps a second copy (.ALT) and attempts to switch to it when damage is detected.[9] This does not always work, requiring manually reverting to an older copy of the registry. [10]
- The registry does not document itself in the same way a configuration file can, since it does not contain comments.
- Restoring parts of the registry is hard because the user cannot easily extract data from backed up registry files. Offline reading and manipulation of the registry (for example from a parallel installed Windows or a boot CD) is not trivial (but not impossible).
- Any application that does not uninstall properly, or does not have an uninstaller, can leave entries in the registry. In some cases this leads to performance or even stability problems, but only if the application registers itself as a class in HKEY_CLASSES_ROOT or HKEY_LOCAL_MACHINE. Note that user settings usually remain in the registry, which is done by design for three reasons: First,

the user might be on a Windows domain with server-based profiles, where the settings move with the user to other computers. Uninstalling the application on one computer does not mean the user does not want to use the program on some other computer on the domain. Second, the uninstall process would have to load and process all user's hives to be able to erase all of its saved settings; to do so for every user can be very processor and time consuming and is not guaranteed to remove from every user in every configuration Windows can be. Third, if the program is installed again later, the user's previous settings will remain. (In any case, unused keys in HKCU have negligible impact on system performance.)

- Since at least 1998,[11] pages at Microsoft Support relating to editing the registry include the disclaimer that the use of the Registry Editor should be done at one's own risk, underlining the severity of a corrupt registry.
- Applications that make use of the registry to store and retrieve their settings are not conductive for use on portable devices used to carry applications from one system to another. Since the settings are in the registry, and the registry is not on the portable device along with the application, any setting changes are lost and must be re-entered for each new system. A similar problem presents itself if the user gets a new computer or needs to reinstall Windows itself; it is difficult to isolate the portions of the registry that should be moved so that the user can retain their settings.

Windows 9x OS

On Windows 9x computers, an older installation can have a very large registry that slows down the computer's startup and can make the computer unstable. This has led to frequent criticisms that the registry leads to instability. However, as the on-disc structure of the registry is entirely different on the NT line of Operating Systems (including Windows XP and Vista) than Windows 9x series OS,[1] slowdown due to registry bloat now occurs much less frequently.

Equivalents in other operating systems

Other systems use separate configuration files for separate application subsystems, but group them together for ease of management. For instance, under Unix and Linux, system-wide configuration files (information which would appear in HKEY_LOCAL_MACHINE on Windows) are traditionally stored in files in `/etc/` and its subdirectories, or sometimes in `/usr/local/etc`. Per-user information (information that would be in HKEY_CURRENT_USER) is stored in hidden directories and files (that start with a period) within the user's home directory.

Applications running on Apple Inc.'s Mac OS X operating system typically store settings in property list files which are usually stored in each user's Library folder. An advantage of this is that corruption to one of these files will normally only affect a single application, whereas corruption of one of the Registry hives can have system-wide effects. However, Mac OS X also has a system database called NetInfo that stores system-wide settings such as user account details and network configuration.

RISC OS also allows applications to be copied into directories easily, as opposed to the separate installation program that typifies Windows applications. If one wishes to remove the application, it is possible to simply delete the folder belonging to the application.[12] This is possible because RISC OS does not support multi-user environments with different settings for each user.

IBM AIX (a Unix variant) uses a registry component called Object Data Manager (ODM). The ODM is used to store information about system and device configuration. An extensive set of tools and utilities provides users with means of extending, checking, correcting the ODM database. The ODM stores its information in several files, default location is /etc/objrepos.

The GNOME desktop environment uses a registry-like interface called GConf for storing configuration settings for the desktop and applications. However, in GConf, all application settings are stored in separate files, thereby eliminating a single point of failure.

The Elektra Initiative provides an alternative back-end for text configuration files for the Linux operating system, similar to the registry.

See also

- Registry cleaner

References

1. ^ *a* *b* Windows 2000 Registry: Latest Features and APIs Provide the Power to Customize and Extend Your Apps. Retrieved on 2007-07-19.
2. ^ Chen, R. *The Old New Thing*, Addison-Wesley, 2007, p. 322.
3. ^ Registry hives. Retrieved on 2007-07-19.
4. ^ Description of the Microsoft Windows registry. Retrieved on 2007-07-19.
5. ^ Microsoft's *Windows 2000 Security Hardening Guide* version 1.3, published May 15, 2003
6. ^ Reading and Writing Registry Values with Visual Basic. Retrieved on 2007-07-19.
7. ^ REG command in Windows XP. Retrieved on 2007-07-19.
8. ^ Microsoft Corporation
9. ^ Inside the Registry. Retrieved on 2007-12-28.
10. ^ How to recover from a corrupted registry that prevents Windows XP from starting. Retrieved on 2007-12-28.
11. ^ Are Windows NT and WIndows 95 Unsupportable?. Retrieved on 2007-07-19.
12. ^ RISC OS tour. Retrieved on 2007-07-19.

- Russinovich, Mark E.; Solomon, David A. (2005). *Microsoft Windows Internals*, Fourth Edition, Microsoft Press, 183-236. ISBN 978-0-7356-1917-3.

External links

- Windows Registry info.& reference in the MSDN Library
- Security Accounts Manager - low-level registry and SAM information
- De-Mystifying The Windows Vista Registry, By Anand C Khanse, MVP
- The basics of the Windows registry and how to clean it up at HowStuffWorks

Retrieved from "http://en.wikipedia.org/wiki/Windows_Registry"

Categories: Windows components | Configuration files